BTeV Trigger/DAQ Innovations

Margaret Votava on behalf of the BTeV DAQ and Trigger Groups

***Abstract*—The BTeV experiment was a collider based HEP B-physics experiment proposed at Fermilab. It included a large-scale, high speed trigger/data acquisition (DAQ) system, reading data off the detector at 500 Gbytes/sec and writing to mass storage at 200 Mbytes/sec. The online design was considered to be highly credible in terms of technical feasibility, schedule and cost. This paper will give an overview of the overall trigger/DAQ architecture, highlight some of the challenges, and describe the BTeV approach to solving some of the technical challenges.**

At the time of termination in early 2005, the experiment had just passed its baseline review with flying colors. Although not fully implemented, many of the architecture choices, design, and prototype work for the online system (both trigger and DAQ) were well on their way to completion. Other large, high-speed online systems may have interest in the some of the design choices and directions of BTeV, including (a) a commodity-based L1 tracking trigger running asynchronously at full rate, (b) the hierarchical control and fault tolerance in a large real time environment, (c) a partitioning model that supports offline processing on the online farms during idle periods with plans for dynamic load balancing, and (d) an independent parallel highway architecture.

***Index Terms*—Data acquisition, large-scale systems, real time systems, triggering.**

I. Introduction

THE proposed BTeV detector [1] consisted of 6 separate subdetectors: pixel, silicon strips, straw tubes, RICH, EMCAL and muon with the pixel detector dominating the channel count (see Table 1).

Table 1. Channel Count

Subsystem	Channels	DCB Subsystems
Pixel	23M	10
Strips	118K	2
Straws	54K	6
RICH	144K	4
EMCAL	10K	2
Muon	37K	4

The overall detector was designed to run with a 396nsec (2.5 MHz) crossing rate. The timing system was designed to deliver a 7.5 MHz clock to allow for calibration data to be taken between crossings.. See Table 2 for the data rates at each trigger level.

Table 2. Event Rates

	Frequency	Event Size	Data Rate
Into L1	2.5 MHz	200 KB	500 GB/sec
Into L2/L3	50 KHz	250 KB	12.5 GB/sec
Into archival	2.5 KHz	80 KB	200 MB/sec

Error! Reference source not found.shows the overall architecture of the BTeV trigger and DAQ.

There are several points in **Error! Reference source not found.** worth noting. First, the detector was unique in that all the data was brought off the detector and digitized in subdetector-specific front end boards. This data was sent via point to point copper cable to data combiner boards (DCBs) before being sent to the first level trigger over optical links. The DCB design was common across subdetectors[1]. This architecture benefited the design in that 1) much of the L1 trigger could be done in software which allowed for the commodity components described in detail elsewhere [2] 2) the DCBs provided a single entry point into the trigger/DAQ that could be centrally designed and maintained, reducing the long term support load.

Secondly, data collected in the DCBs were routed to 8 independent, parallel highways (see Section ? for a more detailed description of the DCB design). This design reduced the control overhead on each particular highway and grouped data coming out of the L1 buffers into larger packets for better network performance.

Manuscript received June 5,2005. This work was supported in part by the U.S. Department of Commerce under Grant BS123456 (sponsor and financial support acknowledgment goes here).

M. Votava is with the Fermi National Accelerator Laboratory, Batavia, IL 60510 USA (phone: 630-840-2625; fax: 630-840-; e-mail: Votava@fnal.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar. colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

[1] More precisely, the output to the high speed optical links was in common. There were two variants to the input side. One for FPIX front end boards (Pixel and Strip detectors) and one for nonFPIX boards. See page **Error! Bookmark not defined.** for more detail.

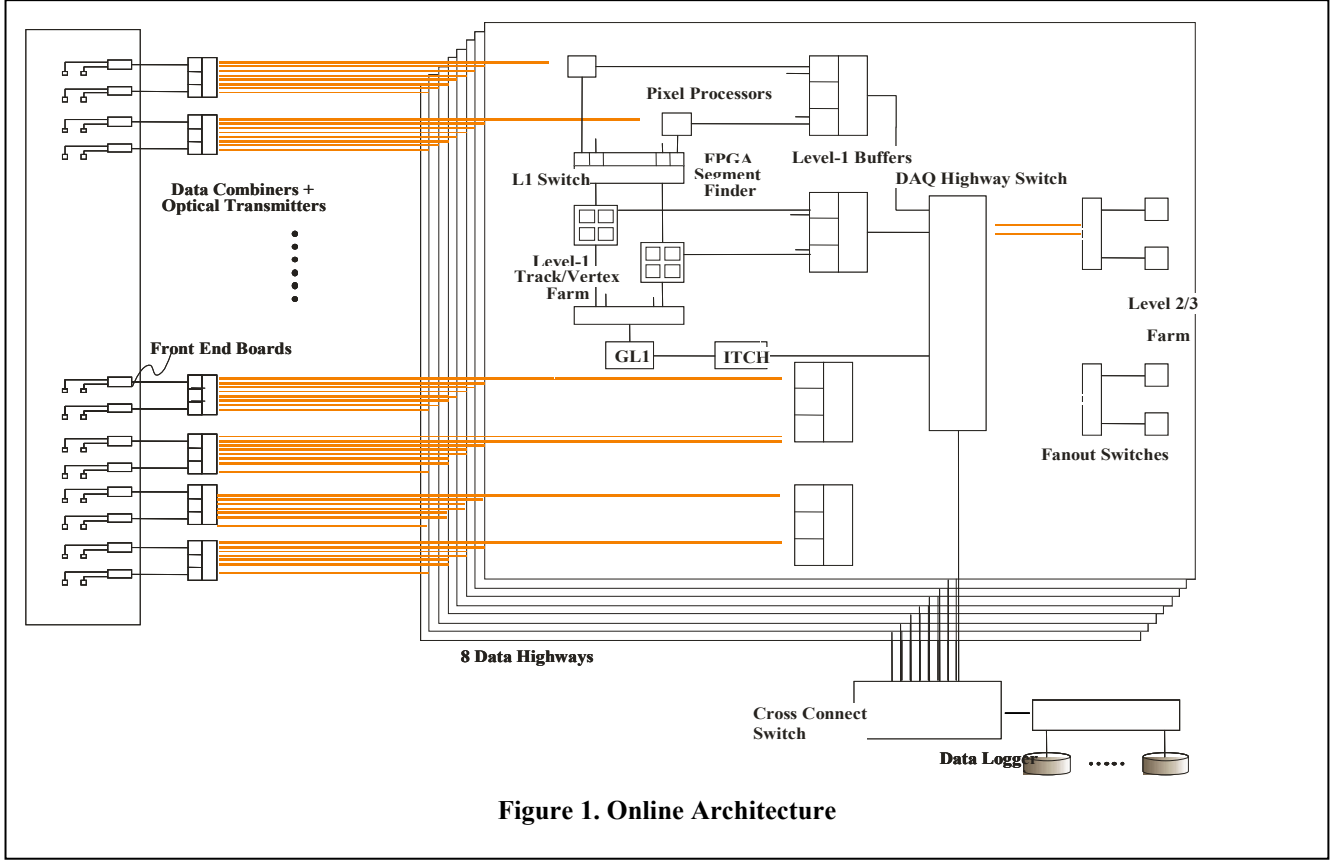


Figure 1. Online Architecture

Thirdly, the Level 2 and 3 trigger (L2 and L3, respectively) decisions were made on the same L2/L3 farm. While the second trigger level was based on pixel and possibly muon data, the third level (L3) used all available subdetector data to reach its decision. The practical difference was the amount of data from a particular event being evaluated and the physics algorithms deployed.

Lastly, the baseline of the BTeV architecture was to log experiment data to large disk farms, ie no tape, using dCache as the underlying data storage support software.

The following sections will spell out some of the features of the online system: highway architecture (II), fault tolerance (III), commodity L1 farm {III}, and partitioning (IV).

II. Highway Architecture

A. DCB

Let's look at the highway architecture in more detail starting with the DCB.

The DCBs were a custom component and the first interface between the subdetectors' front end electronics and the common data readout system. Two flavors of this board existed – one for the FPIX readout (Pixel and Strip detectors) and one for everyone else. For the purposes of this paper, they can be considered the same with only a variation on the configuration of the input ports and rates.

DCBs were the place in the online architecture that split the detector data into highways – eight parallel and independent data streams each processing 1/8th of the detector data. The original DCB design routed a crossing at a time. This resulted in a complex routing table algorithm to factor out any periodicity in the tevatron. The DCB sent data for a given crossing sent to one and only one highway.

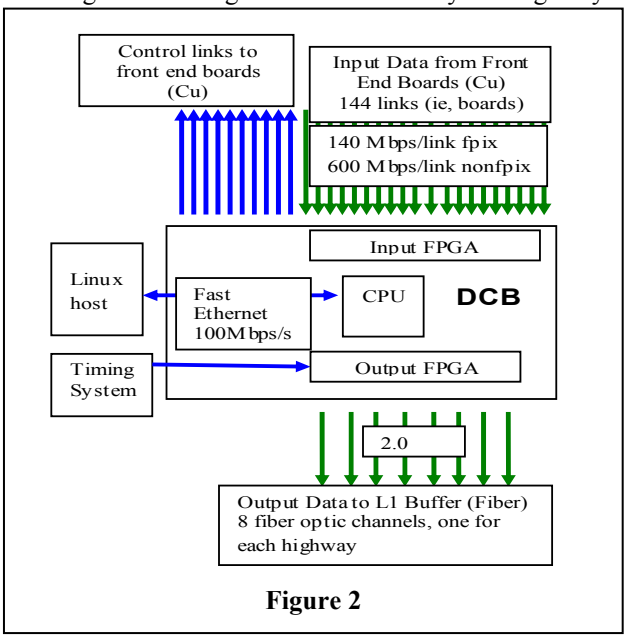


Figure 2

Error! Reference source not found. shows the data and control flow in/out of a DCB. Each board contained a commercial CPU and fast Ethernet interface for control and low speed data readout for diagnostics and commissioning. The DCBs received precise clock information from the timing system We imagined being able to reset/reconfigure "live" on a future crossing. E.g, if a catastrophic highway failure occured, the DCBs could be sent a control command to route to 7 highways instead of 8 starting at crossing number "N".

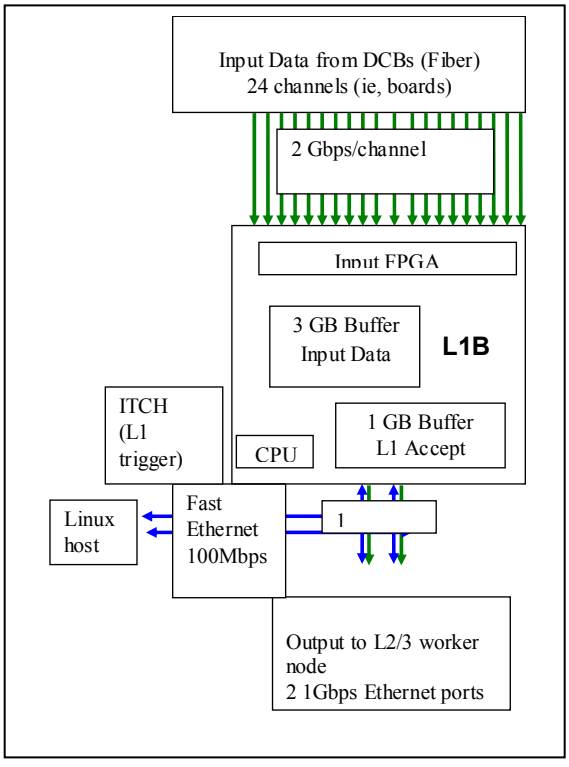
We were investigating the possibility of routing data a turn at a time. This alternative approach would simplify the routing, but increase the length of time data would live in

the DCB, exposing it to a higher single event upset rate. It would also mean that data from FPIX sources would need to be time ordered, a functionality that was being designed into the pixel L1 trigger. Turn routing also had the advantage of being self balancing. That is, the traffic carried over a single highway was the same (on average) for each of the highways. This was true no matter how many highways were in service. If a highway dropped out, the remaining highways would have absorbed the load evenly.

B. *Level 1 Buffers*

Data from the DCBs were routed over point to point optical links into L1 buffers, another custom electronics component. Each L1 Buffer could communicate with 24 DCBs (two crates worth). It's this unit that we will talk about in the partitioning section regarding resource reservation.

For the two detectors that fed into the L1 trigger (pixel and muon), data from the DCBs were also read out by the segment preprocessors. Calculated data and decisions from the L1 trigger would then feed into additional L1 buffers within the trigger.



The primary responsibility of an L1 buffer is to buffer the detector data for a long enough time to make an L1 trigger decision. There were 24 L1 buffers per highway with 3G of memory each for an aggregate total of ~0.5 TBytes of memory, or roughly 1 second of data. Its flow diagram in shown in **Error! Reference source not found.**.

C. *L1 Trigger*

Data comes into the input FPGA and is stored in a circular buffer. When an L1accept is received, all data for that subevent is copied into a smaller 1GB memory area that will be used when transferring accepted events to the downstream L2 trigger farm. No special information is needed from the ITCH (Information Transfer Control Hardware) regarding L1 rejects; the data is simply overwritten in the circular buffer.

This separate L1 trigger accept buffer in memory will help in distributing an event to a second L2 worker node. This capability might be of interest when a specific event satisfies the trigger tables in more than one partition (see section 5). Details need to be flushed out regarding bookkeeping in the L1 buffer to understand exactly when an accepted event can be overwritten (ie, when it has successfully been transferred to the primary partition) while remaining in memory as long as possible to allow for parasitic L2 nodes to transfer additional copies.

D. *L2/L3 Trigger*

From the L1 buffers, data is sent over a GBit Ethernet link to the to the L2/L3 trigger farm The L2/L3 trigger farm is running on the same hardware with the distinction being which physics algorithm would be currently processing the data. An other difference between L2 and L3 is that L2 keeps it's output data in memory while L3 is responsible for the final event building It consisted of commodity processors and was also split into highways. Data from one highway could be physically sent to another highway, but at a price in performance. Each highway consisted of about 90 worker nodes, each with dual CPUs.

To reduce the control overhead and complexity of the software, we designed the event building switch with enough capacity to send a complete event at L2. Each CPU box was referred to as a worker node. Worker nodes would declare themselves to a particular partition, ie, trigger list (see section on partitioning) and notify the ITCH when they were ready for data. The ITCH would assign them a particular crossing number. All data from that crossing would be sent to that worker node.

Worker nodes themselves were grouped into functional units in a highway. Each group was controlled by a regional manager consisiting of 12 worker nodes. A regional manager was responsible for configuring its aassociated worker nodes, fanning out control commands and collecting status, caching DMBS data (eg, various versions of the trigger algorithm), and handling regional faults.

III. FAULT TOLERANCE

The BTeV trigger performs sophisticated computations using large ensembles of FPGAs and conventional

microprocessors. This system will have on the order of 5000 computing elements and many networks and data switches. The need for fault-tolerant, fault-adaptive, and flexible techniques and software to manage this huge computing platform has been identified as one of the most challenging aspects of this project.

As a response to this challenge, the Real Time Embedded Systems (RTES) project group was formed and funded through a 5 year NSF grant. This research group is a collaborative effort between computer scientists and high energy physicists. It is researching the design and implementation of high-performance, heterogenous, reliable, and fault-adaptive real-time systems that are embedded .

because of the overal asychronous nature of this complex system, there is no "hard failure" if a specific computation is not delivered in time, just a temporary degradation of the performance. Either the computation takes a bit longer than expected, or, conversely, the bunch crossing is declared lossed, or "rejected" if indeed the computation takes too long. Such timeouts are expected to occur. In other words, the specification on event rate is always described on a statistical basis.

The RTES project incorporates a multi-aspect solution to large, real-time system modeling and fault adaptation. Model integrated computing methods are used with a graphical modeling environment to describe system integration, messaging types, run control state machines, user interface definitions, and custom ARMOR elements (below), as well as metamodels which define the domain-specific graphical languages for each of these. ARMORs (adaptive reconfigurable mobile objects for reliability) provide intrinsically reliable high level process oversight, with the ability to stop/restart applications in-place (on the same node), or to relocate the work to an available alternative resource. VLAs (very lightweight agents) provide low impact, low level detection of fault conditions.

These aspects have been demonstrated both in a prototype of the BTeV L1 embedded processor farm, as well as in a prototype of the L2/3 commodity computer farm [x].

IV. L1 TRIGGER

One particular highlight of the BTeV design is a commodity based L1 tracking trigger running asynchronously at full rate. Details ?

V. PARTITIONING

Partitioning of the detector was defined to be running multiple independent data acquisition systems in parallel. The value of partitioning, and when to partition the detector are different depending on the phase of the project. I.e., partitioning needs during commissiong (testing the subdetectors in parallel) may be different than when testing new L3 trigger algorithms while taking physics quality data. Partitioning would have also allowed spare cycles on the online trigger farm(s) to be used for other offline processing when beam is off or luminosity low without drastic change to the system configuration. The BTeV L2/L3 farm contained significant processing power. It was an expensive investment to be idle during periods of no beam or low luminosity.

Partitioning is strictly a logical concept. One must map this onto the physical implementation of the experiment. The online DAQ/trigger was constructed in 2 stages to match the proposed funding profile with roughly 50% of capacity at each stage. The first stage consisted of 4 highways with the second stage added the next fiscal year. Even within a stage, individual highways were commissioned one at a time. The logical concept of partitioning needed to support running multiple partitions on a single highway (when only 1 was constructed) as well as the final system with 8 highways

The parallel highway architecture and dynamic reloading of DCB routing tables gave BTeV overwhelming flexibility in this regard, so much so that it became a source of confusion. It's important to note that at the time of BTeV's cancellation, the detailed requirements and design of partitioning were a very hot topic among the online staff and hadn't yet reached buy in from the collaboration. What we will discuss here are some of the ideas.

First, we imagined the cycle of running a partition involved the following steps:

Selecting/reserving [a subset of] electronics to be read out

Defining how much L2/L3 trigger processing power was needed

Initializing the hardware

Collecting the data

Freeing the resources

Original ideas were for a physicist to select the strips and straw front end crates, request 50 Mflops of L2/L3 nodes for processing, and then let software map this out onto a physical implementation. Depending on how many nodes might be needed, the layout may be to run on a single highway or to route to n highways.

An additional constraint on the system was that, because of it's architecture, the L1 trigger hardware on a particular highway could NOT be partitioned; however multiple sets of trigger tables could be loaded into Global Level 1. It was imagined that a given L2/L3 node could belong to one and only one partition.

We had not reached a consensus on the details of partitioning at the time of termination. Because of the sheer number of electronics involved in the Trigger/DAQ, we were converging on the idea of the L1 trigger and active highways always being available as a shared resource. A human run coordinator would establish the overall online configuration for a period of time (day/week/month) and coordinate the individual groups taking data during this period. This stable configuration period had a fixed and predefined set of allowable highways. Subdetector groups still had to select the specific electronics to read out (in units of L1 buffers). These front end crates could be reserved for read/write or readonly (ie, can't be reset or intitialized). It was the responsibility of the run coordinator to schedule the detector so that users could get write access as needed. Partitions could come and go then, adding/removing trigger tables as necessary.

For example, say the run coordinator has made 4 highways available for the next two days. The pixel group could reserve the pixel front end electronics and associated L1 buffers for read/write, and load the pixel trigger table. Crossings would be distributed to all 4 highways. Online software would assign specific L2/L3 nodes to this partition as constrained by the run coordinator. The silicon strip group could come and reserve silicon electronics for read/write and pixel for read only and load a second set of trigger tables. Again, the software would assign L2/L3 worker nodes specifically to this partition. If a given crossing passed the L1 trigger for both partitions, it could be routed to worker nodes in both partitions or split between the two partitions in a predefined scalar. This was also still being discussed in the collaboration.

Partioning became an obvious solution when discussing the problem of how to utilize spare online cycles for offline. The pure Computer Scientists involved in RTES promoted real time scheduling on the worker nodes to squeak out the maximum CPU utilization, but they were outweighed by the opinion that a particular worker node should be single tasking for more simplistic and manageable operation.

Nodes could manually by moved between online and offline partitions, but we also envisioned an automatic shift toward offline as luminosity in the tevatron dropped. An overall luminosity profile would be loaded at the start of a run. As system monitoring tools detected lower utilization in the worker nodes, they would migrate nodes into the offline partition.

VI. Conclusion

Thanks to the efforts of many talented people in the collaboration and at the lab as well as extremely helpful comments from the many reviews, the DAQ and trigger groups in BTeV were able to develop a highly credible online architecture. This architecture was believable in terms of cost and schedule with well understood, and mimimal risks.

Key elements in the success of the design were developing an architecture which maximized the number of commodity components (switching fabric, trigger farms) and minimized the variability between custom boards, i.e, the DCBs were common across all detectors.

References

[1] BTeV Collaboration, BTeV Proposal and TDR. Available: http://www-btev.fnal.gov/public/hep/general/proposal/index.shtml

[2] M. Wang, " A Commodity Solution Based High Data Rate Asynchronous Trigger System for Hadron Collider Experiments", Proceedings of the the IEEE Real Time Conference 2005, Stockholm, Sweden, June 2005, to be published.

[3] M. J. Haney, et al., "The RTES Project - BTeV, and Beyond", Proceedings of the the IEEE Real Time Conference 2005, Stockholm, Sweden, June 2005, to be published.